**TY B.Tech. (CSE) – II [ 2022-23 ]**

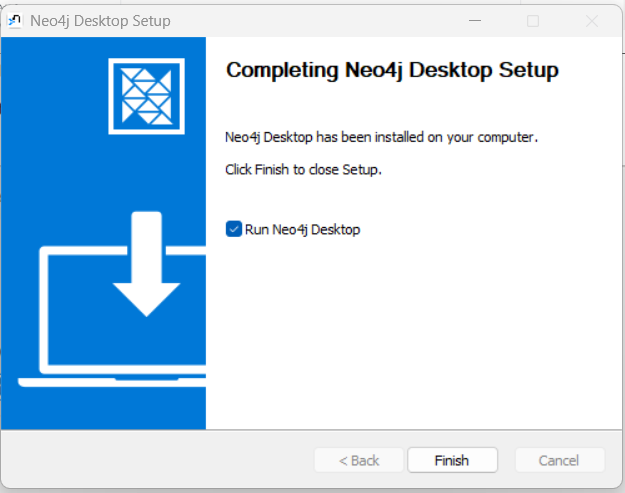
**5CS372 : Advanced Database System Lab.**

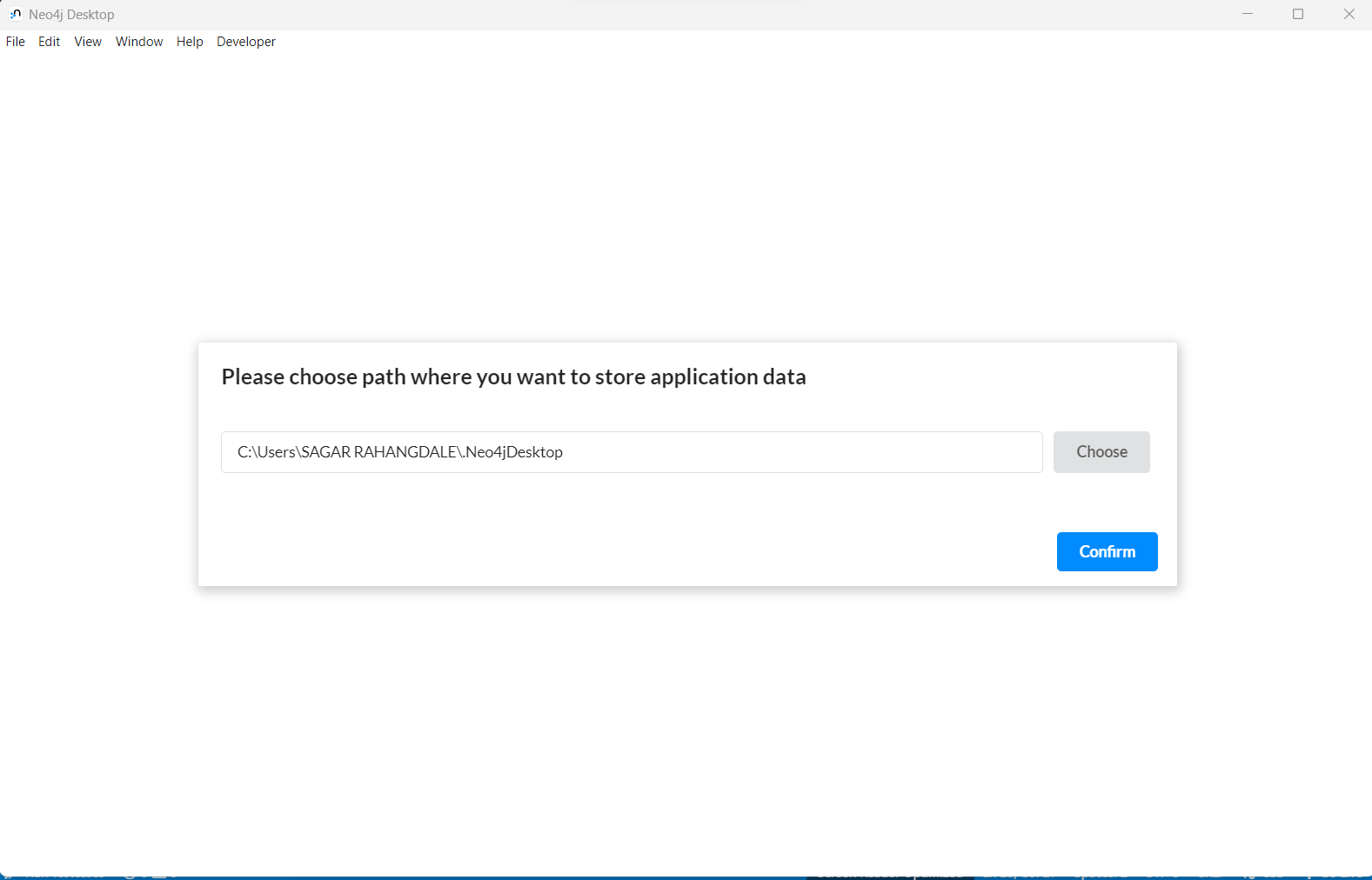
**Assignment No. 11**

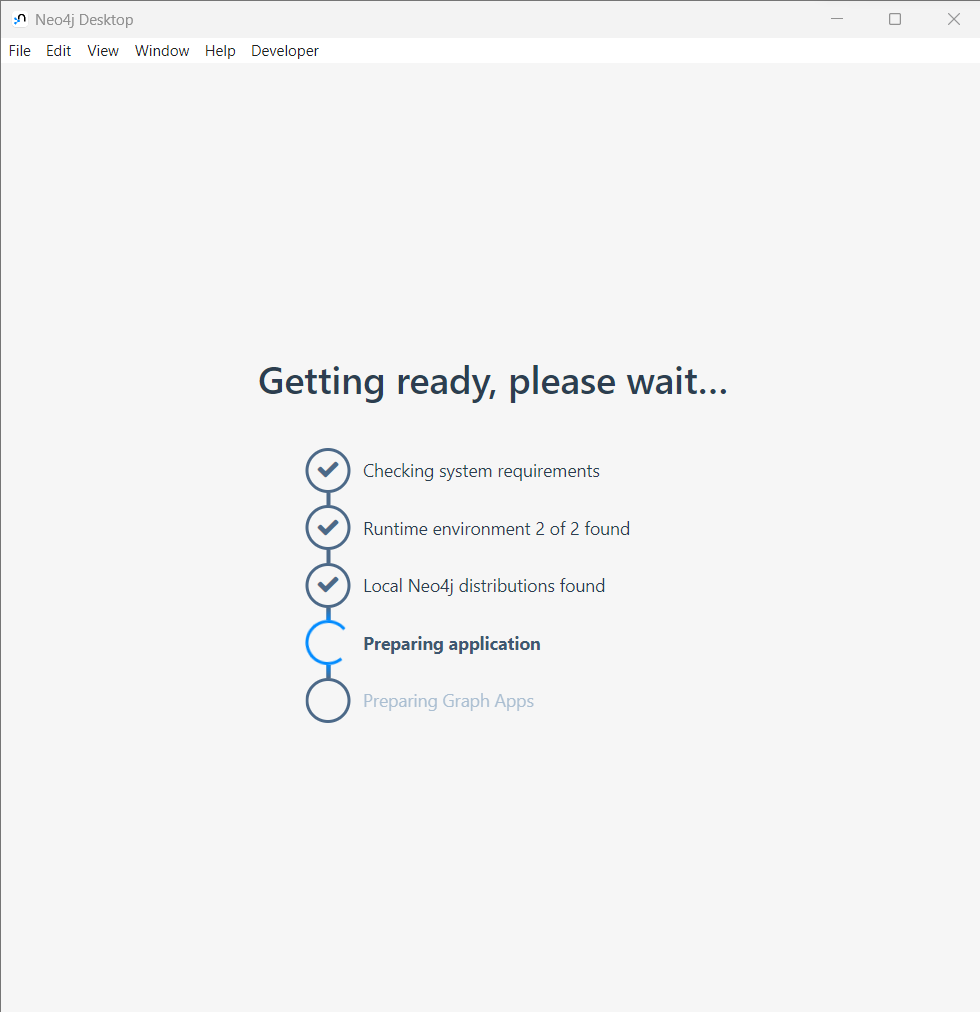
**Neo4j Graph Database**

Consider the “**Research Papers Database”** scenario as follows :

The research papers have authors (often more than one). Most papers have a classification (what the paper is about). The classifications form a hierarchy in several levels (for example, the classification “Databases” has the subclassifications “Relational” and “Object-Oriented”). A paper usually has a list of references, which are other papers. These are called citations.

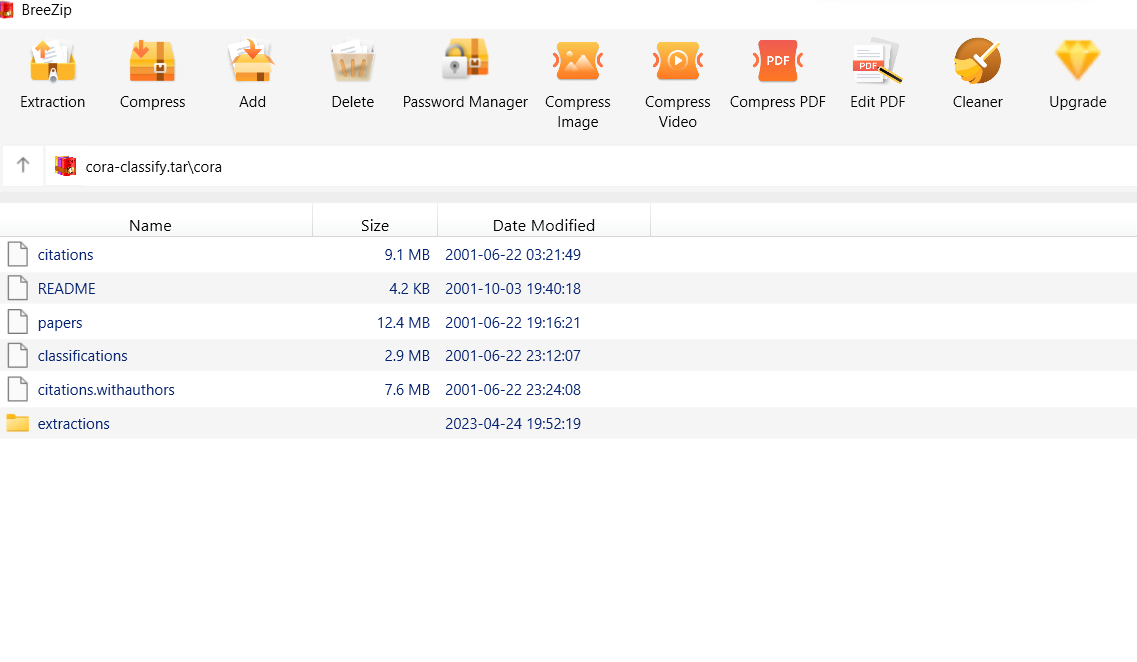






1. Design/model the graph database using Neo4j for above scenario.

1. Download the raw data from **Cora Research Paper Classification Project** : <http://people.cs.umass.edu/~mccallum/data.html>The database contains approximately 25,000 authors, 37,000 papers and 220,000 relationships.



1. Load this data using Neo4j Data Browser

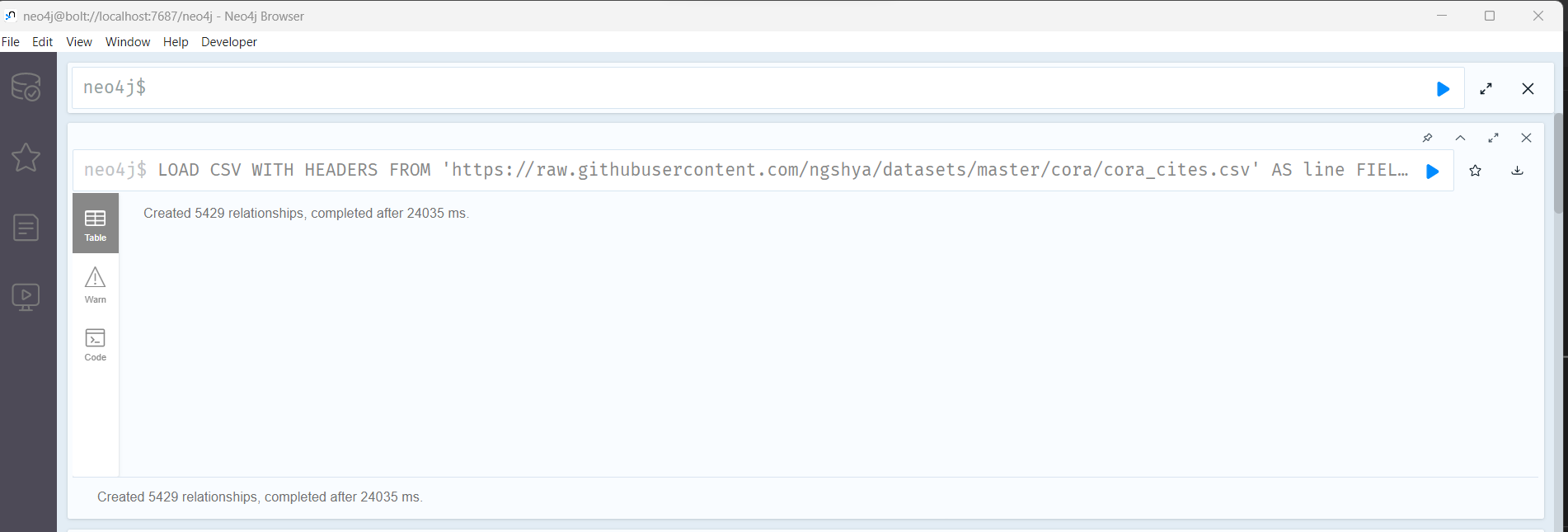


LOAD CSV WITH HEADERS FROM

'https://raw.githubusercontent.com/ngshya/datasets/master/cora/cora\_content.csv'

AS line FIELDTERMINATOR ','

CREATE (:Paper {id: line.paper\_id, class: line.label})



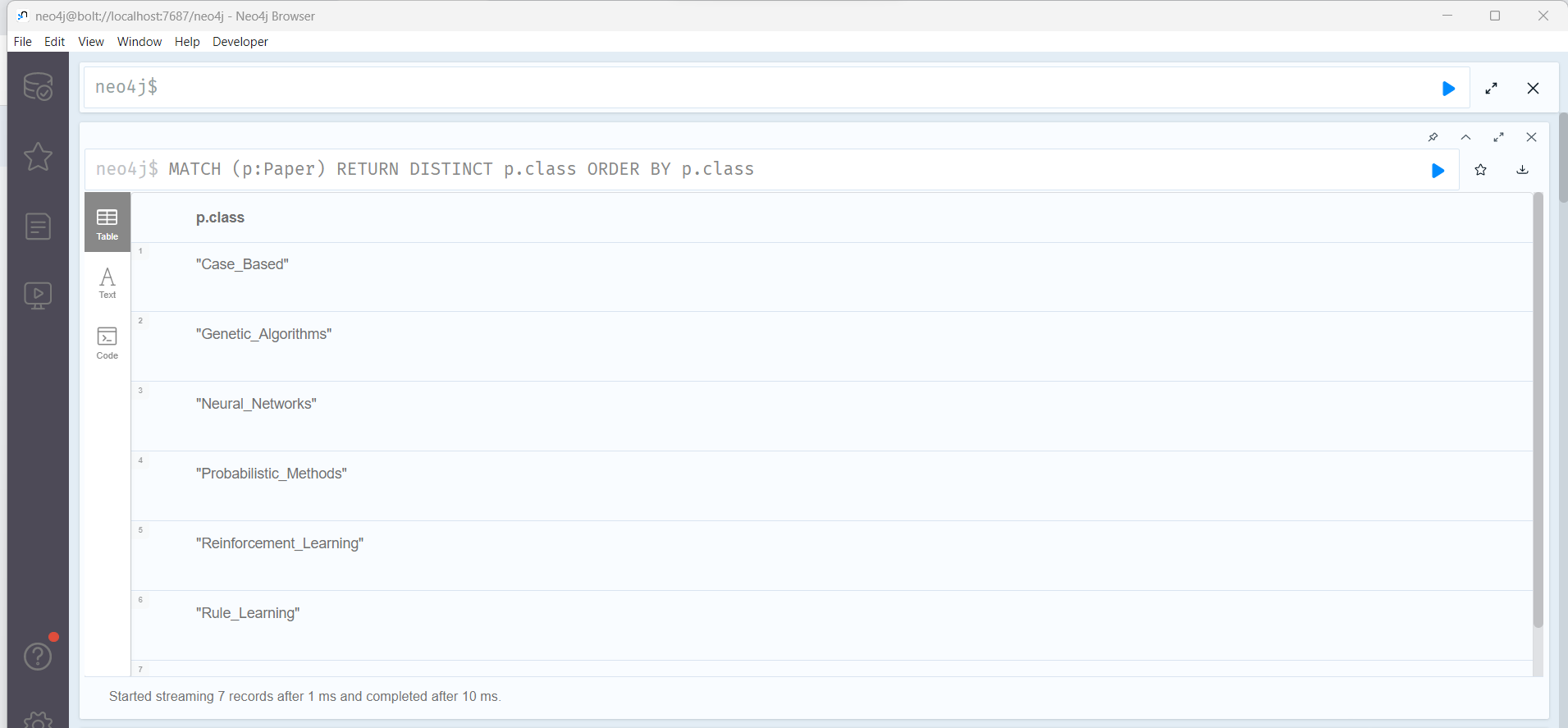
LOAD CSV WITH HEADERS FROM

'https://raw.githubusercontent.com/ngshya/datasets/master/cora/cora\_cites.csv'

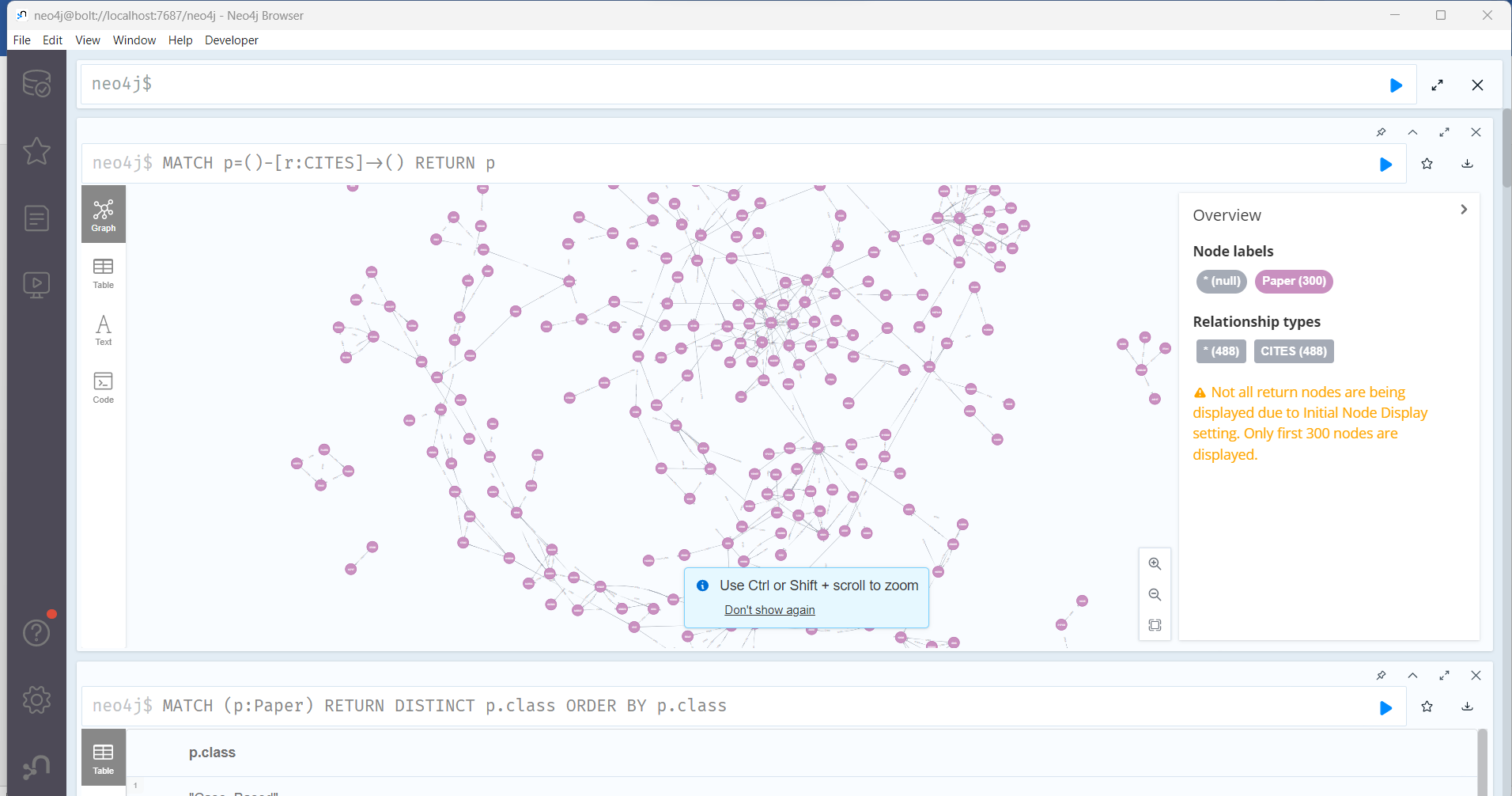
AS line FIELDTERMINATOR ','

MATCH (citing\_paper:Paper {id: line.citing\_paper\_id}),(cited\_paper:Paper {id: line.cited\_paper\_id})

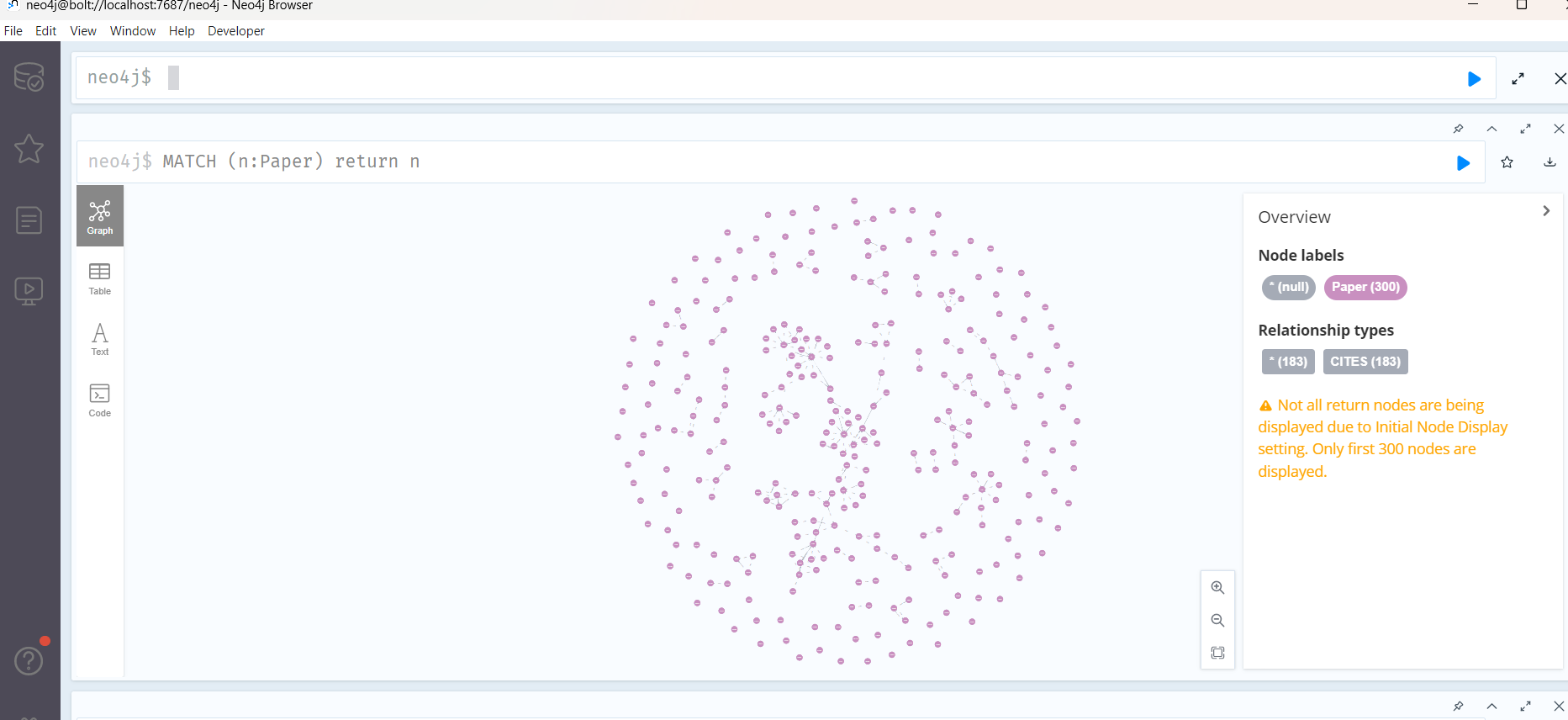
CREATE (citing\_paper)-[:CITES]->(cited\_paper)



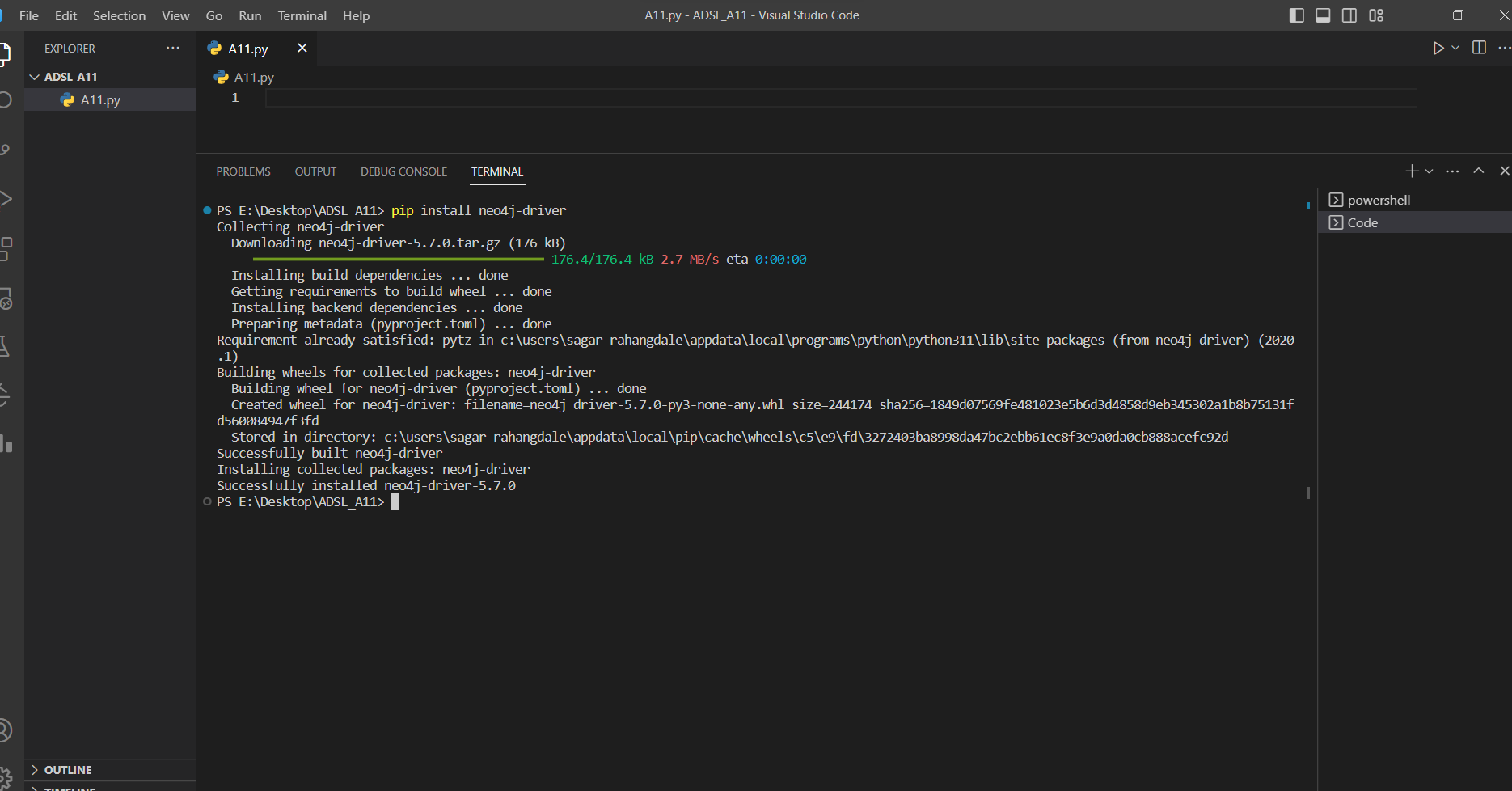
MATCH (p:Paper) RETURN DISTINCT p.class ORDER BY p.class



MATCH p=()-[r:CITES]->() RETURN p

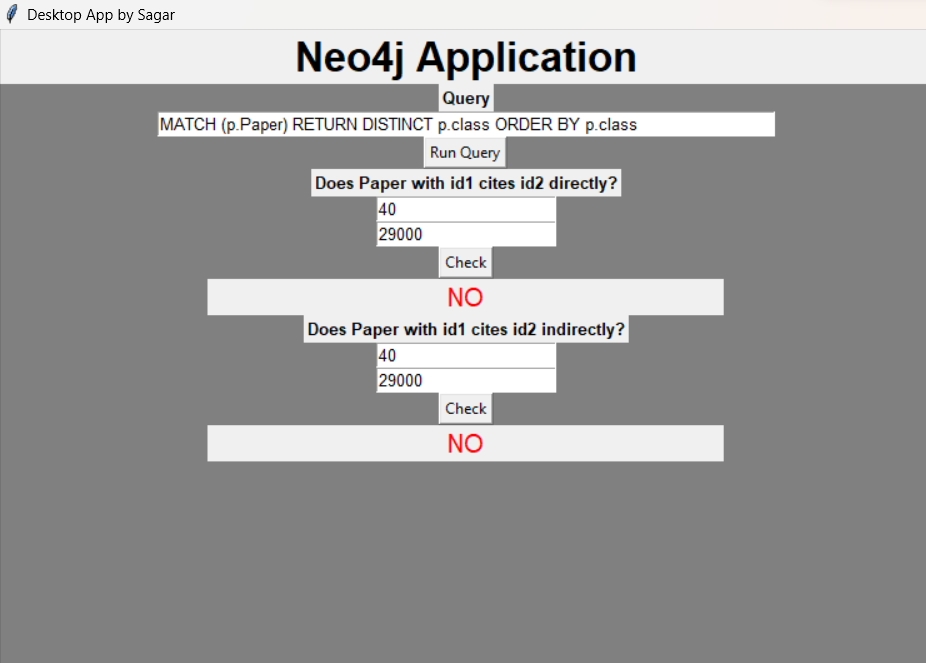


1. Design the python based desktop application for any kind of search on above database. The application should able to answer queries like



* 1. Does paper A cite paper B? If not directly, does paper A cite a paper which in its turn cites paper B? And so on, in several levels.

1. import sys
2. import os
3. import tkinter as tk
4. from tkinter import \*
5. import tkinter.messagebox
6. # For Neo4j Connection
7. from neo4j import GraphDatabase
8. class Neo4jConnection:
9. def \_\_init\_\_(self, uri, user, pwd):
10. self.\_\_uri = uri
11. self.\_\_user = user
12. self.\_\_pwd = pwd
13. self.\_\_driver = None
14. try:
15. self.\_\_driver = GraphDatabase.driver(
16. self.\_\_uri, auth=(self.\_\_user, self.\_\_pwd))
17. except Exception as e:
18. print("Failed to create the driver:", e)
19. def close(self):
20. if self.\_\_driver is not None:
21. self.\_\_driver.close()
22. def query(self, query, db=None):
23. assert self.\_\_driver is not None, "Driver not initialized!"
24. session = None
25. response = None
26. try:
27. session = self.\_\_driver.session(
28. database=db) if db is not None else self.\_\_driver.session()
29. response = list(session.run(query))
30. except Exception as e:
31. print("Query failed:", e)
32. finally:
33. if session is not None:
34. session.close()
35. return response
36. conn = Neo4jConnection(uri="bolt://localhost:7687", user="neo4j", pwd="neo4j")
37. # ^ Neo4j Connected
38. window = tk.Tk()
39. window.title("Desktop App by Sagar")
40. window.geometry("700x500")
41. window.configure(bg="grey")
42. blog = tk.StringVar()
43. blog\_title = tk.StringVar()
44. direct\_id1 = tk.StringVar()
45. direct\_id2 = tk.StringVar()
46. recur\_id1 = tk.StringVar()
47. recur\_id2 = tk.StringVar()
48. # submitting query
49. def submit():
50. query\_string = blog\_title.get()
51. result = conn.query(query\_string, db='neo4j')
52. print(result)
53. blog.set("")
54. def direct\_check():
55. id1 = direct\_id1.get()
56. id2 = direct\_id2.get()
57. query\_string = '''MATCH p=(:Paper{id:"'''+id1 + \
58. '''"})-[r:CITES]->(:Paper{id:"'''+id2+'''"}) RETURN p'''
59. result = conn.query(query\_string, db='neo4j')
60. if (result):
61. Label(window, text="YES", fg="blue", font=(
62. "Arial", 15), width=37).grid(row=160)
63. else:
64. Label(window, text="NO", fg="RED", font=(
65. "Arial", 15), width=37).grid(row=160)
66. blog.set("")
67. def indirect\_check():
68. id1 = recur\_id1.get()
69. id2 = recur\_id2.get()
70. query\_string = '''MATCH p=(:Paper{id:"'''+id1 + \
71. '''"})-[r:CITES]->() MATCH q=(:Paper{id:"'''+id2+'''"}) RETURN q'''
72. result = conn.query(query\_string, db='neo4j')
73. if (result):
74. Label(window, text="YES", fg="blue", font=(
75. "Arial", 15), width=37).grid(row=220)
76. else:
77. Label(window, text="NO", fg="RED", font=(
78. "Arial", 15), width=37).grid(row=220)
79. blog.set("")
80. # tkinter window
81. Label(window, text="Neo4j Application", fg="black",
82. font=("Arial", 25, 'bold'), width=37).grid(row=0, column=0)
83. name\_label = tk.Label(window, text='Query', font=(
84. 'calibre', 10, 'bold')).grid(row=70)
85. name\_entry = tk.Entry(window, textvariable=blog\_title, font=(
86. 'calibre', 10, 'normal'), width=70).grid(row=80)
87. sub\_btn = tk.Button(window, text='Run Query', command=submit).grid(row=110)
88. name\_label = tk.Label(window, text='Does Paper with id1 cites id2 directly?', font=(
89. 'calibre', 10, 'bold')).grid(row=120)
90. name\_entry = tk.Entry(window, textvariable=direct\_id1,
91. font=('calibre', 10, 'normal')).grid(row=130)
92. name\_entry = tk.Entry(window, textvariable=direct\_id2,
93. font=('calibre', 10, 'normal')).grid(row=140)
94. sub\_btn = tk.Button(window, text='Check', command=direct\_check).grid(row=150)
95. name\_label = tk.Label(window, text='Does Paper with id1 cites id2 indirectly?', font=(
96. 'calibre', 10, 'bold')).grid(row=180)
97. name\_entry = tk.Entry(window, textvariable=recur\_id1,
98. font=('calibre', 10, 'normal')).grid(row=190)
99. name\_entry = tk.Entry(window, textvariable=recur\_id2,
100. font=('calibre', 10, 'normal')).grid(row=200)
101. sub\_btn = tk.Button(window, text='Check', command=indirect\_check).grid(row=210)
102. window.mainloop()



* 1. Show the full classification of a paper (for example, Databases / Relational)

**Note : Follow the submission guidelines.**

***Deadline : 16/04/2023***

**Dr. B. F. Momin**

**Course Coordinator**